Engineering Design Graphics Journal (EDGJ)
Winter 2013, Vol. 77, No. 1
http://www.edgj.org

ISSN: 1949-9167

Message from the Chair

Nancy E. Study
Penn State Erie - The Behrend College

Some days I get a bit wistful (frustrated?) and think longingly of my board and T-square. Yes, I am one of those whose first introduction to engineering design graphics was when dinosaurs still roamed the earth and CAD was not prevalent. I drew lines with my mechanical pencil on sheets of green paper, erased mistakes with the aid of an erasing shield, and set text size and font with a lettering template – oh indeed I had one of those because my hand lettering quality was dodgy at best, until required to learn how to letter properly, sans template, in Dr. Larry Drake's descriptive geometry course at Southwest Missouri State University, but that's another story for another time.

So why do I occasionally long for those days? Because sometimes ordinary tasks that should be relatively simple like creating a broken section, modifying the decimal places shown in thread notes, or defining assembly relations, take more time to do, and to demo to my students, than I think they should. That is when I get nostalgic for the simplicity of pencil and paper. When I have one of those days I have even been known to say with all sincerity that I am fully convinced the only difference between one release of any given CAD software and the next is where they place everything in the menus.

What influenced my decision to address this topic in the Message from the Chair was an ongoing discussion with some colleagues on what is, in my opinion, one of the bigger challenges we face by virtue of teaching in a technology rich field. What do we teach our students in the limited classroom time we have? Where is the balance between teaching theory that stays much the same from year to year like rules of multiview projection, dimensioning standards, and visualization techniques, and teaching tool-specific information like solving conflicting constraints, writing code to define relationships between assembled features, creating a self-populating bill of materials, and so on?

Much of the theory I teach my students significantly overlaps what I learned years ago using my trusty traditional tools. So in order to keep software tools from being little more than a really expensive board and T-square, and to teach students how to properly implement the theory in a realistic context, I must learn the ins and outs of those tools. Our students will use these tools, maybe not the exact ones they use in class but similar technology, in their careers. Therefore we must teach not only theory but how to apply that theory with specific tools. Along with that, we must emphasize the importance of adapting to changes in technology and if we expect adaptability from our students, we should expect it of ourselves. I might complain a bit (or a lot) if it takes me a while to find where they relocated the dimension setup menu in the new release of a software, but I

ISSN: 1949-9167

will learn, so my students can learn, and I will continually ask myself, where is that balance between theory and technology?